541,284

10/541284

Rec'd PCT/PTO 01 JUL 2005

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification[7]: G06F 12/02

(21) International Application Number:
PCT/EP2003/014961

(22) International Filing Date:
30 December 2003 (30.12.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
03000125.9   3 January 2003 (03.01.2003)   EP

(71) Applicant (for all designated States except US): AUSTRIA CARD PLASTIKKARTEN UND AUSWEISSYSTEME GMBH [AT/AT]; Lamezanstrasse 2-8, A-1232 Wien (AT).

(72) Inventor; and
(75) Inventor/Applicant (for US only): NASE, Rainer [AT/AT]; Altomontegasse 5, A-1120 Wien (AT).

(74) Agent: RIEBLING, Peter; Postfach 31 60, 88113 Lindau/B. (DE).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
—   without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR BLOCK ORIENTED MEMORY MANAGEMENT PROVIDED IN SMART CARD CONTROLLERS

Structure of the BAT
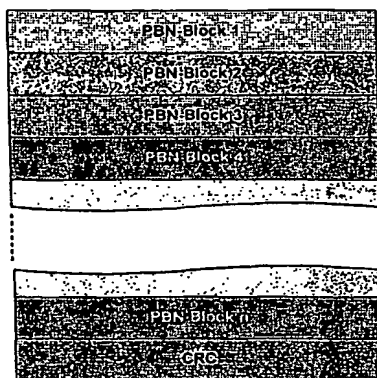
Checksum

(57) Abstract: A method for memory management in smart card controllers by writing of data into a data space in a persistent memory is described. In order to save memory space the persistent memory is split into blocks with fixed data length having logical block numbers; whereby the size of blocks is selected such that it corresponds to the physical size of the pages of the EEPROM memory existing on the card. Fragmented blocks are used by comprising segments being independent from each other, whereby these segments are part of different data fields. A Block Allocation Table (BAT) is used in order to distinguish the physical place of the block in memory from the logical block number and to address different segments in the block.

WO 2004/061673 A2

Best Available Copy

METHOD AND APPARATUS FOR BLOCK-ORIENTED MEMORY MANAGEMENT PROVIDED IN SMART CARD CONTROLLERS

TECHNICAL FIELD

5   The present invention relates to transaction Support in Computer Systems with constrained resources. More specifically, the present invention relates to the minimisation of the amount of memory and the number of write- operations required for transaction support in a resource- constrained environment such as a smartcard.

10

WO 01/71506 A1 offers a method for identification and collection of memory sections that contain information not needed anymore, commonly referred to as garbage.

15   US 6,272,607 B1 describes a method for transactional writing of data into a persistent memory using an intermediate buffer. A method for saving memory space by using a memory management system is not disclosed.

BACKGROUND OF THE INVENTION

20   In smartcards, the computing time, communication bandwidth, and transient memory (RAM) are limited. In addition, writing to persistent memory is much more expensive than writing to temporary memory, and finally no assumptions about the integrity of the communications infrastructure can be made.

25   A problem is that systems with state information residing in a persistent memory such as EEPROM are required to ensure that this information stays consistent even in case of unexpected power losses and other failures.

A transition from one consistent system state to another may involve updates of
30   several cells of persistent memory. These updates should be performed atomically, where either all memory cells are updated or none of them.

If power is lost or some other system failure occurs prior to the completion of the transaction-commit operation, all conditional updates are discarded.

In a resource-constrained environment, such as a smartcard, the size of a
5   transaction buffer is highly critical. Its better to find an implementation, where a transaction buffer isn't need at all. Writing persistent memory is time-consuming compared to reading. The other goal is to reduce the number of expensive write-Operations to persistent Storage needed for transaction support.

10 · Therefore, it is an object of the invention to provide a method and a device for Block-Oriented Memory Management provided in Smart Card Controllers in order to avoid memory waste and time by using a transaction buffer. It is another object of the invention to have the consistency of transactional writing guaranteed using a new update method.
15

SUMMARY OF THE INVENTION

The memory managed by the 'Memory Manager' (MM) is grouped into equal memory blocks having logical block numbers. In an appropriate manner the size of
20   the blocks is selected as such that it corresponds to the physical size of the pages of the EEPROM memory existing on the card.

Based on the selection of this size the time for the writing of data within a block is independent of the number of written bytes. To avoid waste of memory by unused
25   remainders of blocks so-called fragmented blocks are used apart from normal whole blocks, which fragmented blocks comprise segments being independent from each other. These segments are part of different data fields.

A memory is made available only through the formation of memory blocks linked
30   with each other.

Block Allocation Table (BAT)

In order to calculate the physical place of the block from the logical block number a Block Allocation Table (BAT) similar to FAT in DOS is used. Depending on the amount of RAM being available the BAT may either be maintained in the non-
5   transient memory (EEPROM) or in the RAM. If it is maintained in the RAM it must always be set up again when starting the card.

However, it is an advantage of the RAM version that in the course of the further operation of the system, those time consuming writing cycles into EEPROM for
10   updating the BAT are omitted.

Furthermore, the block orientation offers the possibility to implement a secure write mechanism.

15   If data included in a block are updated the block including the new data and those unchanged data of the old block is copied first of all to a new block. Based on the selection of the block size described above copying of the old data does not result in an essential loss of time compared with the writing of the data alone. As long as there are free blocks available desired data may be changed as such without
20   losing the possibility to reject all changes again.

To define the point of no return as exactly as possible, where the new data shall be taken over definitely a bit field having so-called commit bits is created. A commit bit is also included in the header of each block.
25

Only if the commit bit included in the block and the commit bit included in the commit field are identical with each other the block is valid. Based on the placing of the corresponding bits in the commit field the shift from old to new blocks is carried out. To switch to validity of a physical block only the corresponding bit in
30   the commit filed has to be toggled.

In ideal circumstances all commit bits of the commit field are included in one
EEPROM page. By this all validity changes of the blocks are carried out in one
step.

5   The memory management may start now to carry out the necessary changes in
the BAT, cancel those old blocks which are no longer required and supply those
empty blocks to the pool again.

If this work is interrupted by a power breakdown (removal of the card) it may
10 always be finalized after a new start owing to the commit bits provided.

RAM BAT
The usage of the RAM as a memory location of the BAT ensures an acceleration
of write and read accesses in comparison with a BAT written in the EEPROM.
15 Each and every logical block which moves to another physical address must have
a new entry in the BAT. With the corresponding expenditure of time this is lasting
for one write cycle per BAT block.

In accordance with the invention it is provided now that blocks being often updated
20 are not kept in the EEPROM.
Such blocks are kept in the RAM and therefore and with considerable less
expenditure of time they must be updated in the RAM only. If the card is started up
the RAM is initialized so that the RAM BAT is set up there, and on further
operation of the card the RAM BAT is updated only.
25

However, this is one variant of the invention only. Another variant provides that the
chip used is having an essentially greater RAM memory so that the whole BAT
may be kept in the RAM completely and has not to be written in the EEPROM as
is the case in case of the first variant.
30

Fragmented Blocks
It is assumed that a data object is 65 bytes long and that the maximum possible
block length is 60 bytes only. Therefore it is necessary to divide the data object
into two blocks, i.e. one block having 60 bytes which will be filled completely.

However, the other block having a width of 60 bytes is filled with 5 data bytes only. The greatest part of this block filled incompletely would be empty. To avoid this the invention provides for the use of fragmented blocks.

5   If fragmenting was not allowed the remaining 55 bytes existing in the block would remain empty and unused. The usage of fragmented blocks offers the possibility to occupy partial areas in a total block (segments). For this purpose the block header is extended and, at its beginning, has a list with entries providing information to localize the segments as well as their length.
10

By this a 60 byte wide whole block is divided (fragmented) into individual segments, and each segment is belonging to a different data object.

The corresponding segment is identified through the block number of the whole
15   block and the number of the individual segment. In this way the segment can be clearly identified and assigned. Therefore this results in the possibility to group (split) the whole block into individual segments.

Hardware Orientation and Secure Write
20   The other aspect to use a block-oriented system is the better orientation towards the structure of the hardware. The EEPROM is structured in so-called pages, the size of which is in nowadays hardware between 16 and 128 bytes. As an example there are 64 bytes wide pages existing. This is the reason that the block having a size of 64 bytes has been selected, and the advantage to use a page is that it may
25   be deleted and written in common. In general, every write access within one page requires the same execution time, because these actions are carried out by the hardware logic as processes accomplished in parallel.

It is preferred that the width of blocks is equivalent to the width of a EEPROM
30   page or is equivalent to an integer part of this width.

A write access to a 64 bytes page is always carried out at the same time and independent of the fact how many bytes are written on this 64 bytes page. Therefore the write accesses are optimized in accordance with the invention.

5   A further advantage when using pages having a fixed length in the EEPROM is that an improved secure write mechanism may be applied.

If not all data have been written completely it is always possible when storing the data in a memory to go back to the position existing before the data have been
10   written. Therefore this is an atomic access to the memory in a way that all or nothing is being written.

In the prior art the problem was solved as such that first of all data which are required to be written are completely written in an EEPROM buffer. Following the
15   complete writing a sign/symbol is added to the buffer saying that all data have been written now. Only then the buffer may be copied to its final location. Thereafter the buffer is deleted. US 6,272,607 B1 uses an intermediate buffer which is not required in present invention.

20   If there has been an interruption meanwhile, for example if the card is removed, it is always checked if there is something left in the buffer which has not been written yet. It will be read and written as long as the whole contents of the buffer are definitely at that location, where they belong to. Then the buffer is deleted. This is the known function of secure write in accordance with the prior art known.
25

However, it has been recognized by the invention that this effort is unnecessary, because there is a double writing action necessary in accordance with prior art.

First of all there is writing in the buffer, and then the buffer is copied in the course
30.   of a copying process and written in the memory again. This double writing process is avoided in accordance with the invention.

In accordance with the invention the organization is carried out in individual blocks, and therefore individual memory blocks may be replaced by each other.

If data are written somewhere which are existing within one block whole blocks are

5   replaced only then. Then the whole block is rewritten again. Because a logical block organization is existing the block being in the logical row of the blocks is replaced, and this one will be replaced by the original block.

It is not necessary to copy the data, but the old data which were in the old block

10   and are not subject to this updating process are copied into the new block and the new data are written and added at the same time. Then there will be a write access to one block having 64 bytes (if the block size is 64), which write access is always lasting for the same period of time, and double writing in one block is unnecessary.

15

The subdivision into blocks is made, because individual blocks may be very well replaced in the course of an update. At the same time the old block is still available, i.e. at any time desired and if there is an interruption the old block may be activated again and there is a new block, where the new data will be prepared.

20   If all new blocks have been written the new blocks will be declared as being valid and the old blocks may be forgotten only then. Change-over from the old to the new blocks is carried out in the course of the updating process.

Advantages

25   -   Inherent secure write;
    -    efficient formation of atomic writing accesses to different data fields with the possibility of roll back and roll forward;
    -    fragmenting of the memory is avoided to a great extent and limited to fields within fragmented blocks only. If necessary, it is relatively easy to create a

30   garbage collection for fragmented blocks.

Serial Organization of Blocks in Form of a Ring List

There are so-called free blocks existing which are used again and again for the acceptation of data. Also, in the course of this updating process a free block must be found first of all, where the data may be written in. For this purpose a reservoir

5   of free blocks is required. In order to have quick access to the blocks these are organized in form of a ring list.

In accordance with the prior art known it was known to provide those free blocks with an initial identification and store it. Starting from this initial identification action

10   the chain of free blocks was addressed continuously. However, it is a disadvantage of this known method that the initial identification must be stored on the occasion of each access desired. If there is a change a new indicator must be stored at the beginning.

15 · However, if, in accordance with the invention, the chain of the blocks it put in a ring there are essential advantages resulting thereof.
In case of a cold start of the card a free block has to be found in the ring list once only, and therefore all blocks are available, because they are positioned in the ring. Therefore storage is not required, where the ring starts. When the card is

20   started the whole memory is scanned once and checked for consistency. On this occasion a free block is searched for. If it has been found its address is stored in the RAM.

This is similar to fragmented blocks which are arranged in a ring as well. Even

25   here if a new memory segment is required the list is scanned to find out where there is a free segment existing. With the help of this list the management of fragmented blocks is very easy. Therefore it is not necessary to remember the beginning of the blocks, because it is unimportant which block is selected from the ring, because there is access to all blocks organized in the ring automatically.

30

Therefore in accordance with the invention those free and fragmented blocks are organized in the form of a ring.

The Commit Block

In the event of an update copies are made of the original blocks with the new data.
In the case of a data object comprising several blocks there are data existing
being in a block and which are required to be updated.

5   A copy of the block is made which includes the new update data as well.

This block is existing twice now. This means that the block having this logical block
number is existing twice; on the one hand in the form of the old block and on the
other hand in the form of the new block which has been prepared already to
10   replace the old block completely.

It is necessary now to define the moment, where the old block is replaced by the
new block. Therefore a commit block is a block, where such bits are provided only
which are assigned to the blocks.

15

In that moment, where these two bits correspond to each other (e.g. both are 1 or
0) this block is a valid one. The new block set up now is written as such that
compared with the bit stored in the commit block a reverse commit bit is assigned
to it first of all.

20

In that moment, where an updating process is demanded from an old block to a
new one the commit block is rewritten.
Then the bit from the former old block is put in reverse order to what it is itself in
this block, and the bit from the new block is put as such that it corresponds to it.

25

In that moment, where the commit block is newly written as a whole within one
page many blocks desired may be marked as being valid or non-valid at the same
time.

30   DESCRIPTION OF THE DRAWINGS.
Examples of the invention are depicted in the drawings and described in detail
below by way of example.
It is shown in

Fig. 1 schematic view of the BAT

Fig. 2 structure of memory block

5   Fig. 3 structure of fragmented memory block

.Fig. 4 structure of commit block

Fig. 5 chain of free block organized as a ringlist

10

Fig. 6 structure of a logical block pointer.

Fig. 7 flow structure of update process

15   In figure 1 is depicted the structure of MM (Memory Manager) which uses a block-
oriented memory structure. All blocks have the same length and may be identified
by their logical block number (LBN). To resolve the LBN to a physical block
number (PBN) and the physical address, a block allocation table (BAT) (as shown
in figure 1) is used.

20

The BAT may be partly comparable to the FAT in MSDOS. As the block length is
fixed there may be huge amount of wasted memory if the data size doesn't fit to
multiple of the data block size. To avoid this without losing the advantages of a
system with fixed block size, there are so called fragmented blocks which can
25   contain several independent memory segments, belonging to different logical data
units.

To build data units bigger as the block size the blocks may be linked. The linking is
done on a logical level by writing the LBN of the following block to the header of
30   the leading one. A data unit may consist of several (including zero) complete
blocks and one or zero fragmented blocks. It's not possible to link several
fragmented blocks within a data unit. If there is a fragmented block in a data unit
it's always at the end of the unit.

The MM has three layers of memory access. From the view of the MM the upper application layer only knows logical block descriptions (logical block numbers). These are transformed to physical block descriptions with a table. From these the addresses in the memory can then be calculated.

5

The MM is provided with an inherent 'Secure Write'. This means all write or memory accesses are traceable and cancelable until their final validity.

As shown in figure 1 and just explained above the structure of BAT is constructed

10   as a sequence of logical blocks. The entries consist of the physical block numbers. This means that the physical block number (PBN) referenced by a LBN n is to be found at the nth entry. The PBN represents the physical memory address. In figure 2 is depicted the structure of a memory block. Each block consists of a block header and a data area. Besides the LBN of the block itself and LBN of a possibly

15   following block the header contains a 4 bit block ID and a commit bit.

The block ID is used one the one side to identify the type of the block itself and on the other side to identify the segment of possibly following fragmented block. The coding of the section ID:

20

0 :   Block is active (used), not fragmented, no following block or following block
         is not fragmented

1–13 :   Block is active (used) not fragmented, next Block is fragmented,
            value is index of the segment

25   14 :   Block itself is fragmented

15 :   Block is free

The commit bit is used to identify the validity state of the block. If it's value is identical to the value of a corresponding bit in the so called commit field (will be

30   described in detail later on) the block is valid, if not the block is no longer valid and may be erased and included to the list of free blocks.

The block size is determined by:

- The used platform (the block size should not be larger than the EEPROM page size and in case it is smaller it should consists of a whole number conversion)
- The memory size
- The data structure of the applications on the card

5

A standard page size of 64 bytes for each block is assumed. This includes the actual data as well as the overhead from the header size. This leads to a size of 60 bytes for the actual user data.

10   In figure 3 structure of a fragmented block is depicted which is based on that of the normal block in figure 2. The ID of a fragmented block id '0xE'. The block includes an extension located in that area which will be the data area of a normal block. Here additional information is stored, which allows to split of the data are into different independent segments. It starts with a byte for the amount of free space

15   in the block. Following is a list of segment sizes with an entry for every segment in the block.

In figure 4 a structure of a commit block is depicted. The commit field consists of single bits each assigned to one memory block. The position of the bit

20   corresponds to the PBN of the memory block. Only if the bit in the commit and the commit bit in the header of the corresponding block are identical the block is failed. If not the physical block is marked to be erased and be included to the list of free blocks.

25   Typically all commit bits of the field are located in one EEPROM page (a commit block). So writings and updates to the EEPROM memory done before updating the commit field can be switched to be valid within on single write operation on EEPROM. From the moment on the commit field (/commit block) is written a roll forward can be executed even if a power failure occurs. If the power failure occurs

30   before a roll back will be performed. So it's possible to guarantee atomically of write operations even done in many single write cycles. This procedure together with the later described update operation implies a secure write mechanism.

To prevent the system from loosing a valid commit field (respectively commit block) if a power failure occurs during updating the commit block, the commit block is doubled. Only one of the two commit block is valid at a time. An update of the commit block is always done by writing to the block not written to at the last

5   update, because this is the invalid commit block. To distinguish the valid from the invalid block a 2 bit counter is added at the beginning of each commit block (C0,C1). The block with the higher counter value is the block written last, except one of the to counter has the value 11b and the other has the value 00b, than the block with the counter 00b is the last written one.

10

Figure 5 shows an example for a structure of free memory blocks organized in an ring list in order to simplify the handling of free and fragmented blocks. Both lists are organized as ring list. So it's not necessary to store the start of such a list in non volatile memory. It's enough to find a member of such a list on startup

15   procedure and store it in RAM. It's necessary to have these list if new memory must be allocated.

Figure 6 discribes the structure of a logical block pointer (LBP), which allows to identify data objects by their start block and segment (in case of fragmented

20   block). Part of the LBP is the LBN.

As mentioned above a LBP contains a LBN with additional information to reference a data unit with user data. The main difference between LBNs and LBP is that LBNs are used to reference whole memory blocks and LBPs are used to

25   reference the start of data within memory blocks. The upper 4 bits in a LBP represent the segment index of the data area in a fragmented block if a segment in fragmented block should be referenced. If these bits are all zero data in a not fragmented (normal) block is meant.

30   The symbols in figure 6 have the following meaning:
    ID0 – ID3 :   ID of the fragmented area, a part of the address
    LBN :   <u>Logical</u> Block Number (= phy. BN, if memory type = ROM)

Note: To really access specific data within memory blocks an additional offset is needed. This offset bases at the start of the first block in a block chain.

Physical Block Numbers

5 The transformation of logical block numbers to physical block numbers is done by the MM via the BAT. Physical block numbers are only used by the MM to calculate the physical memory address. In this way the BAT can be reduced to the absolutely necessary size.

10 *Update operation*

The update process of data (figure 7) within the memory blocks is designed to guarantee a 'secure write' mechanism. To reach that the data which should be updated isn't touched until the new data is committed. The update data for a block is written together with the unchanged data of the block to a new (former free)

15 block. As the time for writing to a physical EEPROM page is nearly independent of the amount of data bytes, no noticeable delays will be produced by the copying the unchanged data to the new block (see figure). If the update process has finished the operation my be committed by writing a new commit field. After this the old data blocks may be erased and the BAT is updated so that the physical blocks for

20 the concerned logical blocks are exchanged. Figure 9 shows the flow of an update process as described above.

Write operation

Similar to update, but only free blocks are allocated no old data is copied and no

25 blocks are erased after committing. Blocks keep their LBN -> no BAT update necessary.

Delete operation

No operation until commit, then blocks are marked in the commit field as false by toggling the corresponding bit. After commit the blocks are erased and included to

30 the list of free blocks. Blocks keep their LBN -> no BAT update necessary.

CLAIMS

1. A method for memory management in smart card controllers or similar restricted hardware environment by writing of data into a data space in a persistent memory, said method comprising steps of:

a) splitting the persistent memory into blocks with fixed data length having logical block numbers (LBN);

b) selecting the size of blocks as such that it is equal to - or equivalent to an integer ratio of - the length of a page in EEPROM to the physical size of the pages of the EEPROM memory existing on the card;

c) providing a Block Allocation Table (BAT) in order to calculate the physical place of the block in memory from the logical block number;

2. The method according to claim1, including the step of splitting a whole block into individual segments, whereby each fragment is belonging to a different data object.

3. The method according to claim 2, including the step of identifying a corresponding segment through the block number of the whole block and the number of the individual segment.

4. The method according to claim 2, including defining a block header in the block with a list of entries providing information to localize the segments as well as defining their length.

5. The method according to claim 1, wherein a linkage between blocks by writing the LBN of the following block to the header of the leading block is provided.

6. The method according to 1, wherein a secure write mechanism is accomplished by replacing individual memory blocks by each other.

7. The method according to claim 1. wherein some kinds of blocks are organized in form of a ring list.

8. The method according to claim 1 including the step of defining a bit existing in
5   each block header, whereby this bit corresponds to a bit existing in a commit block.

9. The method according to claim 8, where toggling of a bit in the commit block toggles the validity of the corresponding memory block.
10

10. A device with a persistent memory and a block structure comprising:

a) a memory managing system using a block-oriented memory structure;

15   b) blocks with the same length and identifying them by their logical block number (LBN);

c) a block allocation table (BAT) to resolve the logical block number to a physical block number (PBN) and its physical address;
20

11. A device according to claim 10, further comprising a linkage between blocks by writing the LBN of the following block to the header of the leading one.

12. A device according to claim 10 characterized by blocks with fixed block size,
25   which can contain several independent memory segments, belonging to different logical data units.

13. A device according to claim 10 characterized in that the BAT is held in persistent memory (EEPROM).
30

14. A device according to claim 10 characterized in that the BAT is held in non-persistent memory (RAM-BAT) and re-initialized on startup.
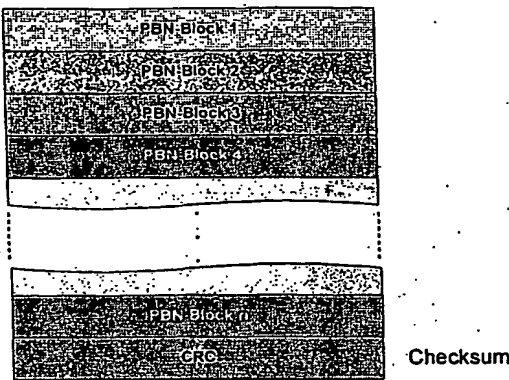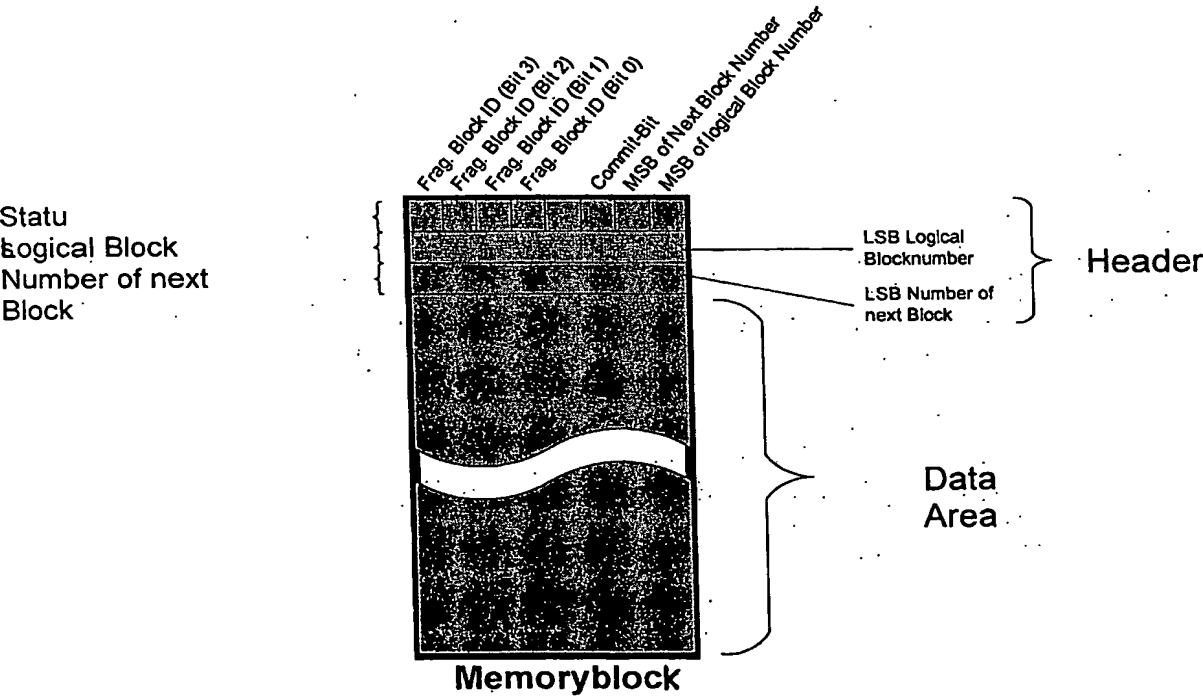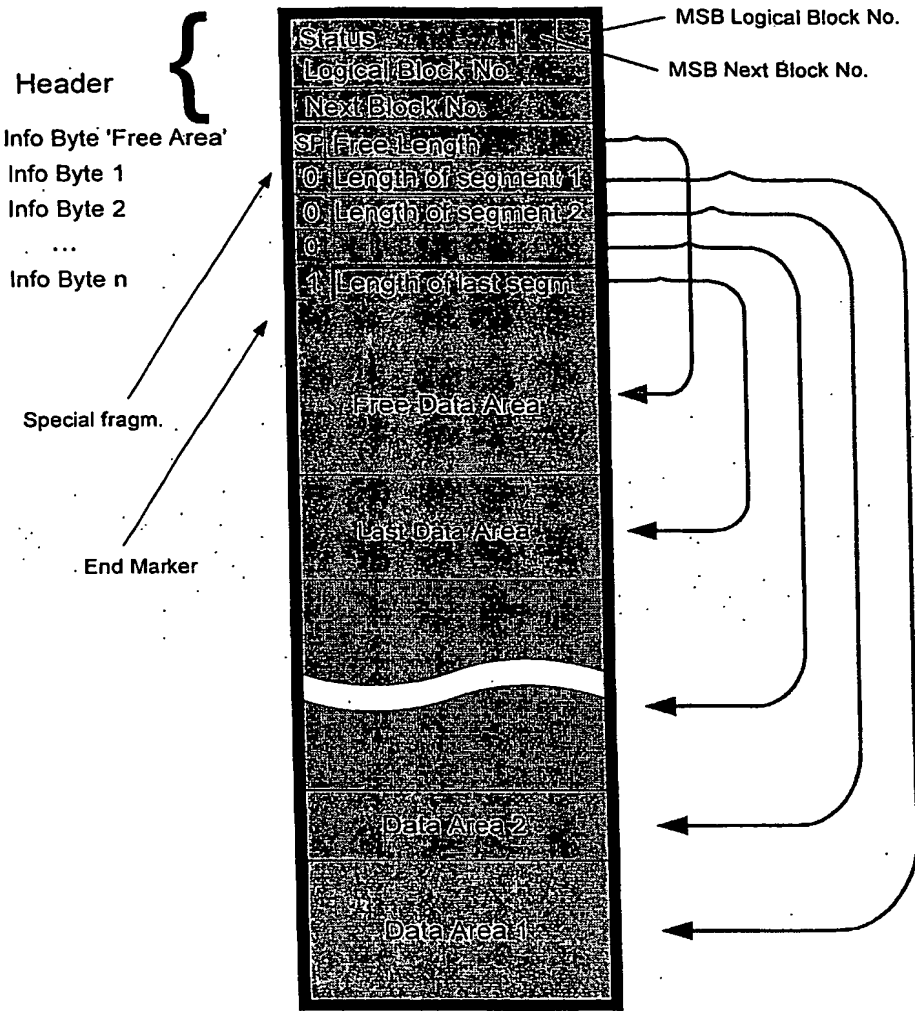
Structure of the BAT

PBN Block 1
PBN Block 2
PBN Block 3
PBN Block 4

PBN Block n

CRC ——— Checksum

Figure 1

Frag. Block ID (Bit 3)
Frag. Block ID (Bit 2)
Frag. Block ID (Bit 1)
Frag. Block ID (Bit 0)
Commit-Bit
MSB of Next Block Number
MSB of logical Block Number

Statu
Logical Block
Number of next
Block

LSB Logical
Blocknumber } Header

LSB Number of
next Block

Data
Area

Memoryblock

Figure 2

Fragmented Memory Block

Figure 3

Byte 0	P7	P6	P5	P4	P3	P2	P1	P0	····· XOR Checksum Start Value '55'
Byte 1	15	14	13	12	C3	C2	C1	C0	····· 4 Bit Counter ('0' is the highest number ... means '10')
Byte 2	23	22	21						

Commit Bit of block
with PBN 505

Byte 63	511	510	509	508	507	506	505	504	····· The position of the bit represents the phys. block number (PBN)
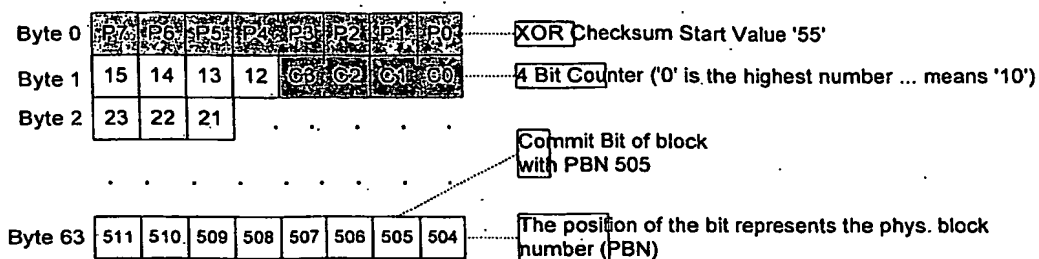
Figure 4

Chain of Free blocks

FreeBlocksRbEndPbn

FreeBlocksCurren

FreeBlocksRbBase = FreeBlocksRbCurrent

Chain of
free Blocks
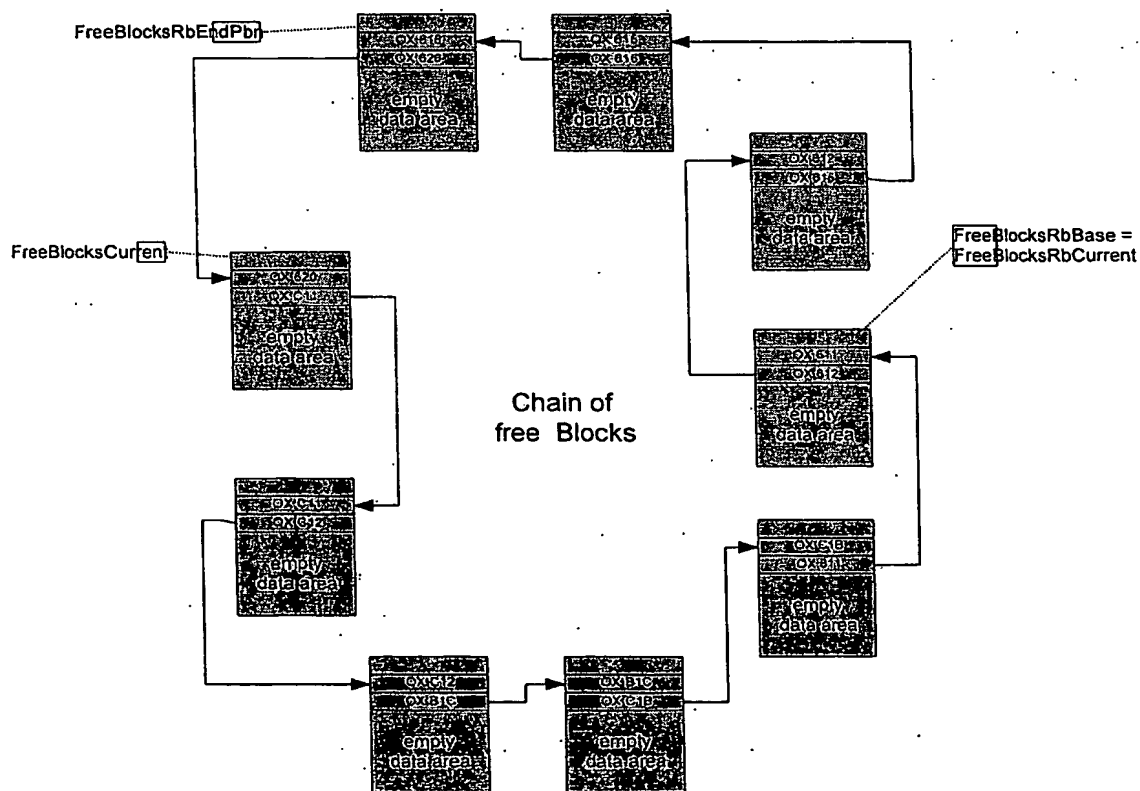
Figure 5

Logical Block Pointer

Bit　15　14　13　12　11　10　9　　　　　　8　－　0

ID
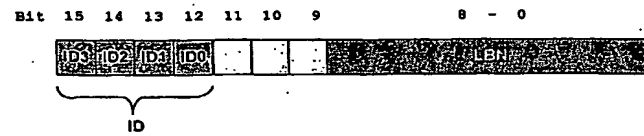
Figure 6

Flow of a Update Process

Figure 7